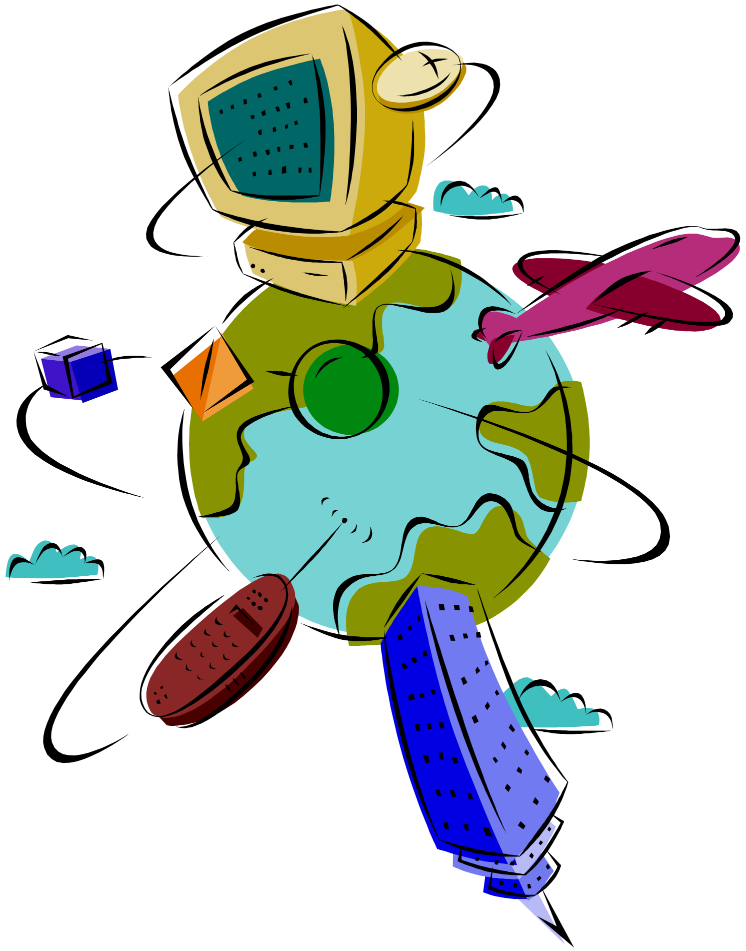
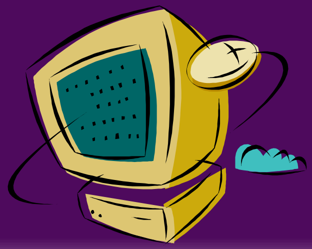


Module 3

CORBA

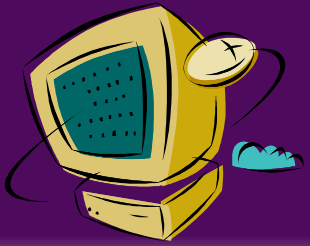
Why We Need CORBA?



Need a solution to develop, deploy, and integrate systems in a distributed heterogeneous environment.

- Diverse OS – Unix, Windows, MacOS etc.
- Diverse Network – TCP/IP, Ethernet, ATM, etc.
- Diverse Programming Language – applications programmed in C++, JAVA, COBOL etc.
- Diverse Hardware Platform.
- Coexist with legacy systems.

What is CORBA?



➢ Architecture for interoperable distributed computing

Developed by the **OMG** (Object Management Group)

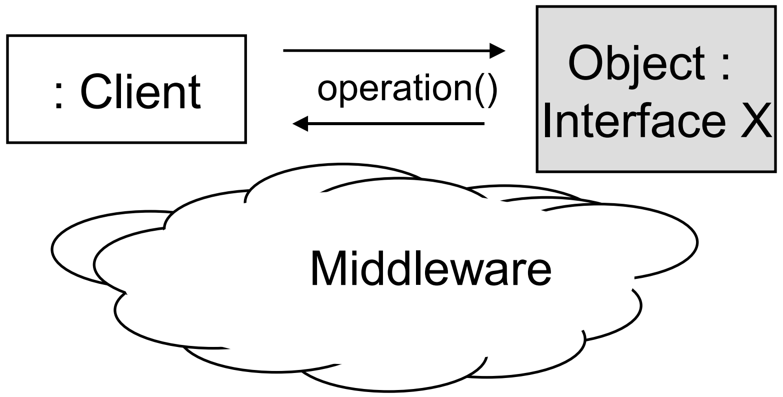
- ✓ A consortium of over 700 companies.
- ✓ **Goal -** to develop, adopt, and promote standards for the development and deployment of applications in distributed heterogeneous environments.

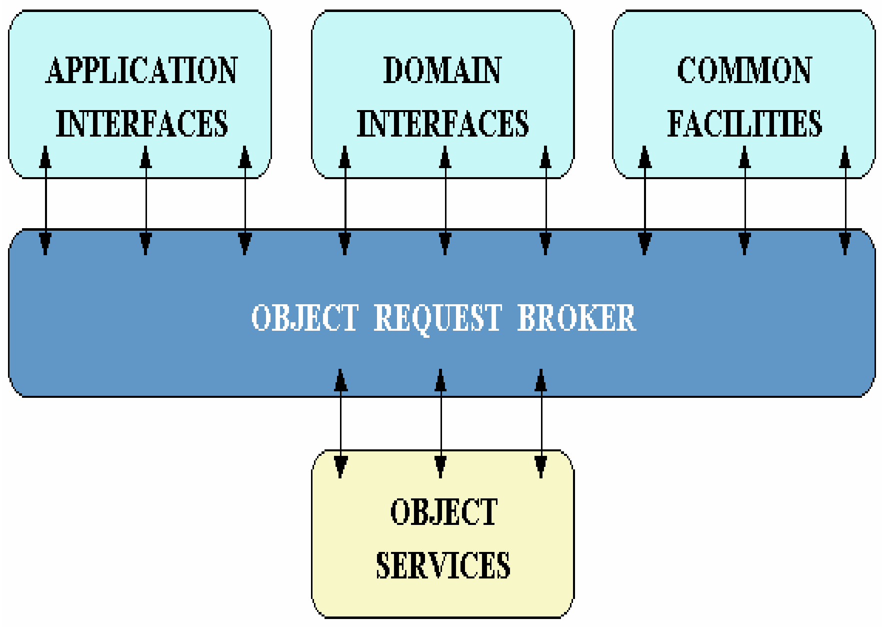
CORBA – Common Object Request Broker Architecture

Middleware that provides the necessary framework and API for developing distributed applications

Middleware :- Middleware is the software that sits '**in the middle**' between applications working on different operating systems

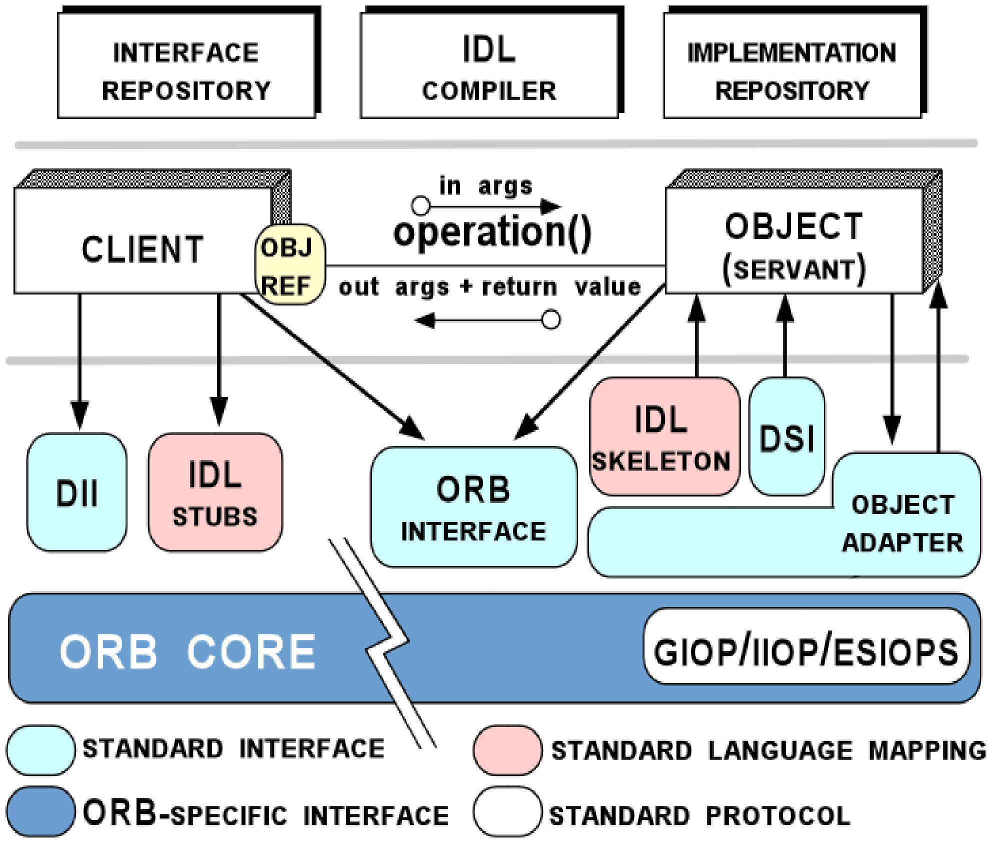
CORBA: **Overview**





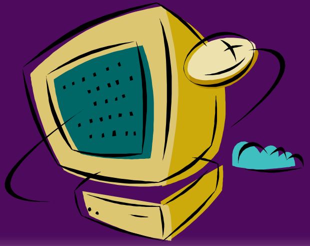
Common Object Request Broker Architecture

Continue……



- **ORB** – *Object Request Broker*
- **IDL**- *Interface definition Language*
- **DII** –*Dynamic Invocation Interface*
- **DSI** – *Dynamic Skelton Interface*
- **GIOP** -*General Inter-ORB Protocol*
- **IIOP** - *Internet Inter-ORB Protocol*
- **ESIOPS** - *Environment-Specific Inter-ORB Protocols*

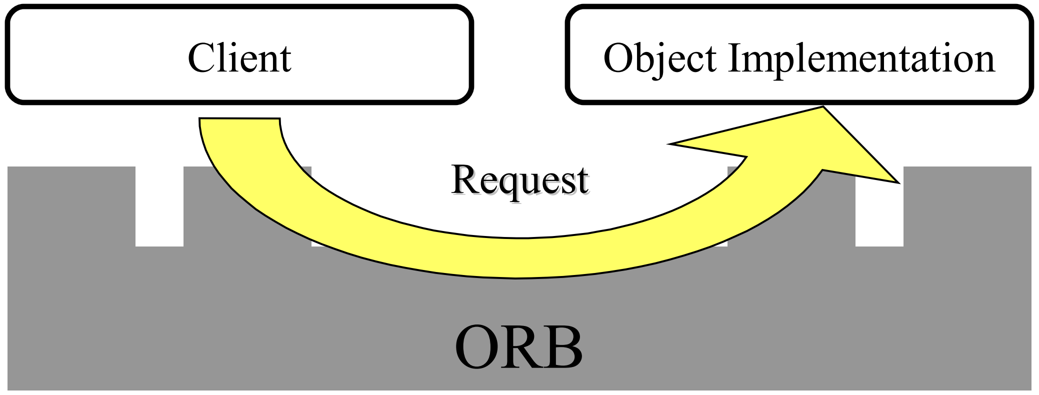
Main Components of CORBA



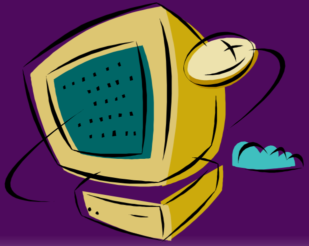
- ORB Core
- CORBA Objects
- OMG Interface Definition Language (OMG IDL)
- Stubs and Skeletons
- Dynamic Invocation Interface, Dynamic Skeleton Interface
- Object Adapter (Portable Object Adapter)
- Interface Repository
- ORB Interoperability

ORB Core

Main Function: To deliver requests to objects and return any responses to the clients making the requests.



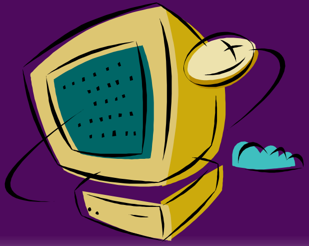
Continue……



Key Feature of the ORB Core

- **Transparent Object Location** – Clients do not know where target objects reside.
- **Transparent Object Implementation** – Clients do not know how objects are implemented. What programming language, or what OS it is running under.
- **Transparent Object Execution State** – Clients do not know if objects are activated or deactivated when making requests. ORB takes care of that.
- **Transparent Object Communication Mechanism** – Clients do not know what mechanism ORB uses to transfer requests, i.e. TCP/IP, local method call etc.

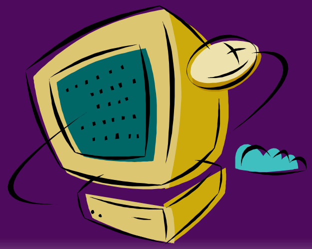
OMG IDL (Interface Definition Language)



An interface is a description of the operations that are offered by an object.

- OMG IDL is a declarative language for defining the interfaces of CORBA objects. All CORBA Objects must be described in OMG IDL.
- The IDL file is then compiled using the special IDL compiler to the specified programming language.
- Currently there are IDL mappings to many popular programming language including JAVA, C, and C++.
- This interface is then implemented using the programming language specified.
- Due to objects having this common interface, inter-object communication is possible.

Stubs and Skeletons



The IDL compiler will generate a client side ***stub*** and server side ***skeleton*** code. Handles the lower network management in distributed applications.

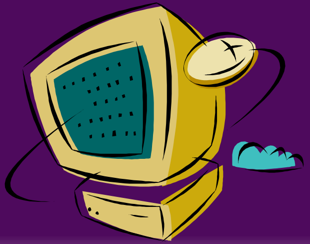
When client invoke an IDL-defined operation on an object reference as if it were a local method, it must link in the <u>stub</u> code.

- Stub is a mechanism that creates and issues request on behalf of the client.
- Stub handles the marshalling of a request, that is converting data structure into wire format.

Once a request reaches the server side, the skeleton code is used to invoke the right method on the right implementation.

- <u>Skeleton</u> code translates wire format data into memory data format (unmarshaling).

Dynamic Invocation Interface (DII)
Dynamic Skeleton Interface (DSI)



The static stub and skeleton codes are limited in use.

DII defines a *request* so that clients who know the

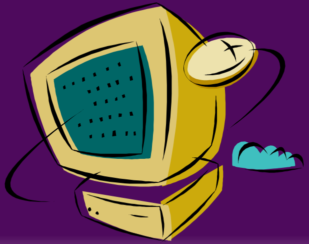
- object reference
- interface type

Can build requests without having to rely on the IDL generated stub code.

DSI deals with requests sent by clients in a generic manner.

Looks at the requested operation and its arguments and interpreting the semantics dynamically.

Object Adapter

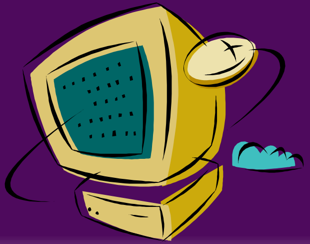


Serves as the glue between CORBA Object implementation and the ORB itself.

Main functionalities of the Object Adapter:

- **Request Demultiplexing** – request arrives for an object, make sure the right servant is located .
- **Operation Dispatching** – once the servant is located, pass along the request to the servant.
- **Activation and deactivation** – activate the object, and deactivate the object when no longer needed.
- **Generating Object References** – generate references to objects for clients to use

Interface Repository



Provides a dynamic mechanism for CORBA-based applications to access OMD IDL type system when it is executing.

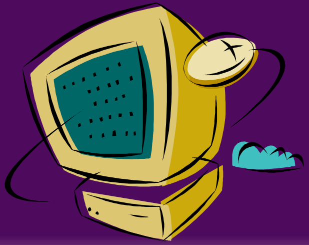
Applications must know the types of values to be passed as request arguments, also applications must know the types of interfaces supported by the objects being used.

•

Most applications know these values statically, because of pre-compiled code. It is necessary sometimes to access these information at runtime

The **IR** can be thought of as a set of objects that encapsulate the IDL definitions of all CORBA types available in a particular domain.

ORB Interoperability



Common Protocols to facilitate transmission of CORBA requests and replies between ORBs.

GIOP – General Inter-ORB Protocol

- ➢ Specified transfer syntax and a standard set of message formats for ORB interoperation over any connection-oriented transport.

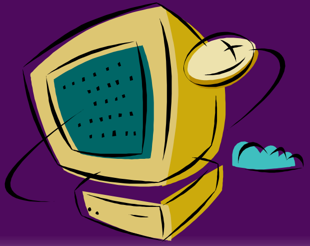
IIOP – Internet Inter-ORB Protocol

- ➢ Becoming standard for most ORB products
- ➢ An implementation of the GIOP for TCP/IP

ESIOP – Environment Specific Inter-ORB Protocol

- ➢ Specific protocol made for already existing distribute computing infrastructure.
- ➢ To bridge to other ORBs, use IIOP.

ORB Interoperability Cont…



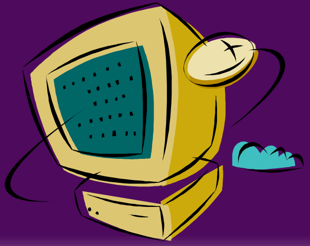
In addition to standard protocols, standard object reference formats are also necessary for ORB interoperability.

ORBs use the contents of object references to help determine how to direct requests to objects.

IOR – Interoperable Object Reference

- Stores information needed to locate and communicate with an object over one or more protocols.
- For example, an IOR containing IIOP information specifically stores hostname and TCP/IP port number information.

How To Program In CORBA



- Write some IDL that describes the interfaces to the object or objects that will be used or implemented.
- Compile the IDL file. This will generates the programming language specific files and also the Stub and Skeleton files.
- Implement the IDL compiler-generated interfaces and classes that are needed for the application.
- Write code to initialize the ORB and inform it of any CORBA objects that have been created.
- Compile all the generated code and our application code with a programming language specific compiler.
- Run the application.